Network Unfolding Algorithm and Universal Spatiotemporal Function Approximation

by D-T. Lin and J.E. Dayhoff

T.R. 95-6

Report Documentation Page					*Form Approved* *OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE **1994**		2. REPORT TYPE			3. DATES COVERED **00-00-1994 to 00-00-1994**
4. TITLE AND SUBTITLE **Network Unfolding Algorithm and Universal Spatiotemporal Function Approximation**					5a. CONTRACT NUMBER
					5b. GRANT NUMBER
					5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)					5d. PROJECT NUMBER
					5e. TASK NUMBER
					5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Institute for Systems Research,University of Maryland,College Park,MD,20742**					8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)					10. SPONSOR/MONITOR'S ACRONYM(S)
					11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT **see report**					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES **23**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Network Unfolding Algorithm and Universal Spatiotemporal Function Approximation*†

Daw-Tung Lin and Judith E. Dayhoff

Institute for Systems Research

University of Maryland

College Park, MD 20742

Abstract

It has previously been known that a feed-forward network with time-delay can be unfolded into a conventional feed-forward network with a time history as input. In this paper, We show explicitly how this unfolding operation can occur, with a newly defined Network Unfolding Algorithm (NUA) that involves creation of virtual units and moving all time delays to a preprocessing stage consisting of the time histories. The NUA provides a tool for analyzing the complexity of the $ATNN$. From this tool, we concluded that the $ATNN$ reduces the cost of network complexity by at least a factor of $O(n)$ compared to an unfolded *Backpropagation* net. We then applied the

*Copyright ©1994 by D.-T. Lin and J. E. Dayhoff. All Rights Reserved.

†This work was supported in part by the National Science Foundation under Grant No. NSF D CDR 8803012 and NSF EEC 94-02384, ,the Naval Research Laboratory (N00014-90K-2010), , and the Applied Physics Laboratory of Johns Hopkins University.

theorem of Funahashi, Hornik et al and Stone-Weierstrass to state the general function approximation ability of the $ATNN$. We furthermore show a lemma (Lemma 1) that the adaptation of time-delays is mathematically equivalent to the adjustment of interconnections on a unfolded feed-forward network provided there are a large enough number ($h2^{nd}$) of hidden units. Since this number of hidden units is often impractically large, we can conclude that the $TDNN$ and $ATNN$ are thus more powerful than BP with a time history.

1 Network Paradigm and Definition

Networks with this capability can play an important role in applications domains that have naturally time-varying properties to their signals and dynamic situations. These domains include identification and control as well as signal processing and speech recognition.

An important contribution in this area has been the time-delay neural network (*TDNN*) proposed by Waibel et al [18], which employs time-delays on connections in feedforward networks and has been successfully applied to speech recognition [19, 9]. The time-delay neural network also classifies spatiotemporal patterns and provides robustness to noise and graceful degradation [14]. Techniques such as backpropagation through time [20] have been applied to temporal pattern recognition but not with adaptive time-delays. Time delays are fixed initially and remain the same throughout training. The decision of how many time delays are needed and what are the most appropriate lengths for each time delay are often made by trial-and-error. As a result, the system may have poor performance due to the inflexibility of time delays and due to a mis-match between the choice of time delay values and the temporal location of the important information in the input patterns. In addition, the system performance may vary depending on the range of the time delay values.

To overcome this limitation, we have used an Adaptive Time Delay Neural Network model [6, 14, 13, 7, 15]. This network adapts its time delay values as well as its weights during training, to better accommodate to changing temporal patterns, and to provide more flexibility for optimization tasks. The *ATNN* used here allows arbitrary placement of time delays along interconnections and adapts those time delays independently of one another. Furthermore, time-windows are not used as in previous work [2, 18] but instead classification relies on a set of individual time delay values associated with each interconnection. Although other artificial neural network architectures that make use of time delays have been suggested [17, 5, 7, 2, 3], these paradigms employ different training rules or different network topologies, and the network presented here is simple to use and has a general formulation.

The proposed *ATNN* model employs modifiable time delays along the interconnections between two processing units, and both time delays and weights are adjusted according to system dynamics in an attempt to achieve the desired optimization. Node i of layer $h-1$ is connected to node j of the next layer h, with the connection line having an independent time-delay $\tau_{jik,h-1}$ and synaptic weight $w_{jik,h-1}$. To illustrate, a three layered *ATNN* is shown in Figure 1.

Next we propose definitions that are used to describe a general *ATNN* architecture with flexible configuration.

Definition 1 *n_{ji} is the number of time-delays employed on the connections to node j from node i (i.e., the number of samples in the time frame $\mathcal{T}_{ji}$).*

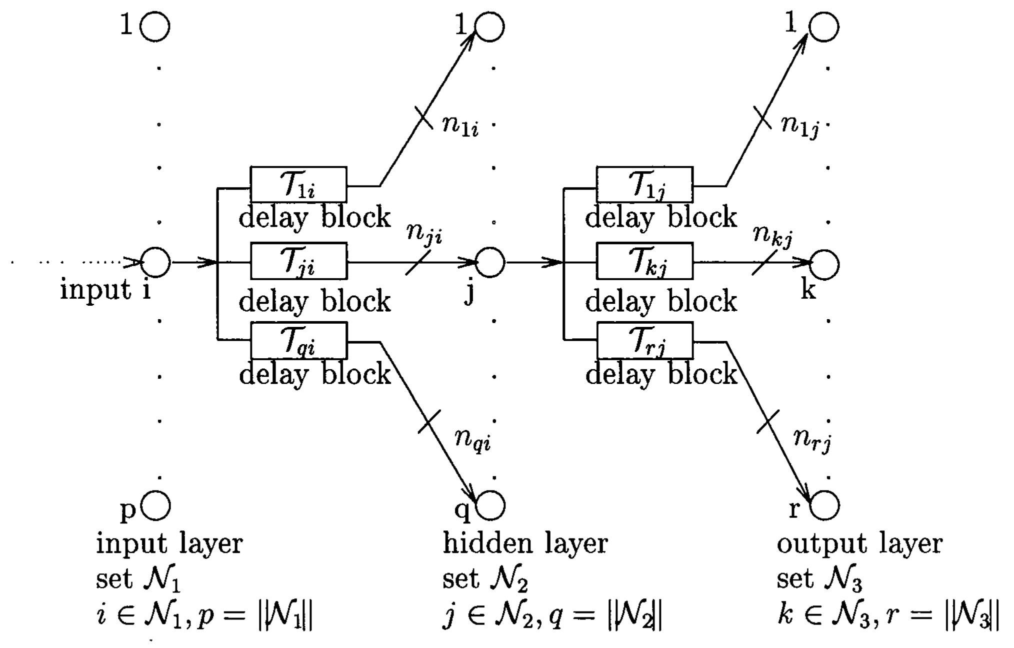

Figure 1: Three layered *ATNN*

Definition 2 Time frame: *Time frame $\mathcal{T}_{ji}$ is a set of time delays $(\tau_{ji1}, ..., \tau_{jin_{ji}})$ employed on the connections to node j from node i. The time frame can have an additional index h $(\mathcal{T}_{ji,h})$ to signify layer h of time-delay.*

Definition 3 *The set of time frames $\mathcal{T}_{\cdot i}$ for connections that originate at node i is defined as:*

$$\mathcal{T}_i = (\mathcal{T}_{1i}, \cdots, \mathcal{T}_{ji}, \cdots, \mathcal{T}_{qi}), \tag{1}$$

where q is the number of units that receive connections from node i.

Definition 4 *$p_{ji}(t, \mathcal{T}_{ji})$ is the set of signal values transmitted to node j from node i via time frame $\mathcal{T}_{ji}$ at time t. Thus, $p_{ji}(t, \mathcal{T}_{ji}) = [s_i(t - \tau_{ji1}), ..., s_i(t - \tau_{jin_{ji}})]$, where $s_i(t)$ is the signal from node i at time t.*

Definition 5 *$P_i(t, \mathcal{T}_{\cdot i})$ is the set of signal values transmitted from node i to other nodes at time t. Thus,*

$$P_i(t, \mathcal{T}_{\cdot i}) = \begin{pmatrix} p_{1i}(t, \mathcal{T}_{1i}) \\ \vdots \\ p_{ji}(t, \mathcal{T}_{ji}) \\ \vdots \\ p_{qi}(t, \mathcal{T}_{qi}) \end{pmatrix} = \begin{pmatrix} [s_i(t-\tau_{1i1}), ..., s_i(t-\tau_{1in_{1i}})] \\ \vdots \\ \left[s_i(t-\tau_{ji1}), ..., s_i(t-\tau_{jin_{ji}})\right] \\ \vdots \\ \left[s_i(t-\tau_{qi1}), ..., s_i(t-\tau_{qin_{qi}})\right] \end{pmatrix}$$

Let $J_{i,h}$ be a set of nodes that receive connections from node i on layer h. Node i transmits the set of pattern values $P_i(t, \mathcal{T}_{\cdot i})$ to selected subset $J_{i,h}$. In other words, $\forall j \in J_{i,h}$, node j receives pattern $P_i(t, \mathcal{T}_{ji})$ from node i through a time frame $\mathcal{T}_{ji}$. The samples in the time frame $\mathcal{T}_{ji}$ correspond to the delays $\tau_{jix,h}$, where $x = 1, 2, \cdots, n_{ji}$, where $n_{ji} = \|\mathcal{T}_{ji}\|, j \in J_{i,h}$, and h is the index of the layer in which i belongs. In the definitions above, we have omitted the layer index h for simplicity.

Definition 6 *$\mathcal{N}_h$ is the set of nodes $\{1, 2, ..., |\mathcal{N}_h|\}$ of layer h*

Thus, each node j of layer h (eg. $j \in \mathcal{N}_h$), receives n_{ji} inputs from each $i \in \mathcal{I}_{j,h}$, where $\mathcal{I}_{j,h}$ is the subset of nodes on layer $h-1$ ($\mathcal{I}_{j,h} \subseteq \mathcal{N}_{h-1}$) that connects to unit j on layer h. The total number of inputs to j is: $m_j = \sum_{i \in \mathcal{I}_{j,h}} n_{ji}$.

In this model, we assume that $p = \|\mathcal{N}_1\|, q = \|\mathcal{N}_2\|, r = \|\mathcal{N}_3\|$ are fixed, and that the connectivity between layers is fixed ($\|J_{i,h-1}\|$ for the same layer is fixed), and for the sake of simplicity, we assume $\|J_{i,1}\| = \|\mathcal{N}_2\|$ and $\|J_{i,2}\| = \|\mathcal{N}_3\|$ (fully connected). In general, the number of samples in the time frame, n_{ji}, and the values of the delays (therefore the set $\mathcal{T}_{ji}$)

are variables. In this research, we assume that n_{ji} is fixed (selected) and allow $\tau_{jix,h-1}$ to be variable. We also assume that n_{ji} is the same for all $i \in \mathcal{I}_{j,h}$ and for all $j \in J_{i,h-1}$. If n_{ji} is the same (fixed) for all $i \in \mathcal{I}_{j,h}$, then $m_j = n_{ji} \cdot \|\mathcal{I}_{j,h}\|$.

Definition 7 *n_{h-1} is defined as the number of time-delays on connections originating at node i on layer $h-1$ where $n_{h-1} = n_{ji}$ for all j.*

Then our model possesses the following property:

Property 1 *For each node $i \in \mathcal{N}_{h-1}$ connecting to all $j \in J_{i,h-1}$, $J_{i,h-1} \subseteq \mathcal{N}_h$, there is a time frame $\mathcal{T}_{ji}$ with n_{h-1} samples, corresponding to the delays $\tau_{jix,h-1}, x = 1, \cdots, n_{h-1}$. Inputs to node j (on layer h) are from node i (on layer $h-1$), and there are $n_{h-1} \cdot \|\mathcal{N}_{h-1}\|$ such inputs.*

2 NUA and Complexity Analysis

From previous literature, researchers have indicated that a time-delay embedded network can be unfolded into feedforward network, but left the procedure of unfolding vague. In this section we elucidate this procedure in a very explicit four-step operation: Network Unfolding Algorithm (NUA).

For notational simplicity, we use a three layer ATNN to illustrate the unfolding operation NUA and to keep the number of time-delays in the same layer to be consistent. We use definitions of the network configuration in previous section. Use Definition 6 and let $n_I = |\mathcal{N}_I|, n_H = |\mathcal{N}_H|, n_O = |\mathcal{N}_O|$ be the number of nodes on the input, hidden and output layers respectively. Use Definition 7 and let $n_1 = d_1, n_2 = d_2$ be the number of delays on connections originating at nodes on the input and hidden layers respectively. From Property 1, there are

$d_1 \cdot n_I$ and $d_2 \cdot n_H$ inputs to each node on the hidden layer and the output layer respectively. Knowing the number of connections, we can unfold the ATNN by the following algorithm:

Algorithm 1 (Network Unfolding Algorithm (NUA)) *:*

```
Step 1 – unfold inputs:
For each hidden node $j$ do in parallel:
    For each input node $i$ do in parallel:
            1.1 duplicate $(d_1 - 1)$ new input nodes and spread these nodes horizontally next to
              each original input node
            1.2 move $(d_1 - 1)$ of the original connections to new input nodes correspondingly
              and retain the weight and time-delay on each connection
    end
end
Step 2 – re-adjust input time lag:
For each hidden node $j$ do in parallel:
        2.1 remove the time-delays between input and hidden units
        2.2 set the input values as $[p_{j1}(t, \mathcal{T}_{j1}), ..., p_{jn_I}(t, \mathcal{T}_{jn_I})]$ accordingly, where $\mathcal{T}_{ji}$ is the time
          frame (in Definition 2) on the connections from node $i$ to $j$, $p_{ji}$ is the signal value
          vector transmitted from node $i$ to $j$ (Definition 4)
end
Step 3 – unfold hidden nodes:
For each output node $k$ do in parallel:
    For each hidden node $j$ do in parallel:
        3.1 duplicate $(d_2 - 1)$ new input nodes and spread these nodes horizontally next to
          each original hidden node
        3.2 move $(d_2 - 1)$ of the original connections to new hidden nodes correspondingly
          and retain the weight and time-delay on each connection to output node $k$
        3.3 for each newly created hidden node do in parallel:
              copy the whole branch which associates with the original hidden node in
              Step 2, and then connect to that new node as its branch and retain the weights
        end
    end
end
```

Step 4 – re-adjust input time lag:

```
For each each hidden node $j$ and its newly created node from Step 3 do in parallel:
    4.1 remove the associated time-delays between hidden and output units
    4.2 re-adjust the input node time lag, such that the input node of each branch takes the
      vectors of $[p_{j1}(t - \tau_{k,j,1}, \mathcal{T}_{j1}), ..., p_{jn_I}(t - \tau_{k,j,1}, \mathcal{T}_{jn_I})], ..., [p_{j1}(t - \tau_{k,j,d_2}, \mathcal{T}_{j1}), ..., p_{jn_I}(t - \tau_{k,j,d_2}, \mathcal{T}_{jn_I})]$ correspondingly
end                                                                                    □
```

Let n_I, n_H, n_O be the number of units in input, hidden and output layer respectively. Let d_1 and d_2 be the number of time-delay connections between input-hidden node pairs and hidden-output node pairs respectively. We present a table (Table 1) which shows how the number of input and hidden units in a feedforward network or BP (with zero weight on connections to other nodes) are increased exponentially from unfolding. If the number of nodes in each layer is n, then the number will be increased by a factor of $O(n)$ and the input nodes is increasing by a factor of $O(n^2)$. Therefore the ATNN is indeed a very economic architecture to achieve the same goal as feedforward network. For a network with more than two hidden layers, the unfold operation can be executed by repeating **Step 3** and **Step 4** from the lowest hidden layer to the highest layer, and expanding the net until reaching the output layer.

Figure 2 explicitly and graphically elucidates the unfold operation of an ATNN with net configuration $2 \xrightarrow{2} 2 \xrightarrow{2} 1$ (labeled as **original**). In **Step 1**, two new nodes $(2 \times 2 - 2 = 2)$ are created (drawn in dotted circle) which are labeled as I_1' and I_2', then spread these two newly created nodes next to its original node. In **Step 2**, each associated time-delay is removed by adjusting the time lag of each input node so that each node receiving information at correct time slot $(t-\tau_1, t-\tau_2, t-\tau_3,$ and $t-\tau_4)$. In **Step 3**, since there are two synapse connections from node H_1 to output node O_1, we need to create another hidden node (labeled in H_1') and then copy the whole branch of node H_1 and hook up to node H_1'. In **Step 4**, the time-delays τ_5 and τ_6 are removed from connections between hidden node and output node. These time lag factors are pushed down to the input layer, so that each branch inherits its value from the upper level. In other words, the input vector of the branch under node H_1 becomes $[x_1(t-\tau_1-\tau_5), x_1(t-\tau_2-\tau_5), x_2(t-\tau_3-\tau_5), x_2(t-\tau_4-\tau_5)]$. The input vector of the other branch (under node H_1') is $[x_1(t-\tau_1-\tau_6), x_1(t-\tau_2-\tau_6), x_2(t-\tau_3-\tau_6), x_2(t-\tau_4-\tau_6)]$.

The NUA can also be executed in an alternative order by unfolding the second layer first using the same procedure. This is illustrated graphically in Figure 2. In **Step 1** and **Step 2**, the hidden layer is unfolded first by creating a virtual node H_1' and duplicating the

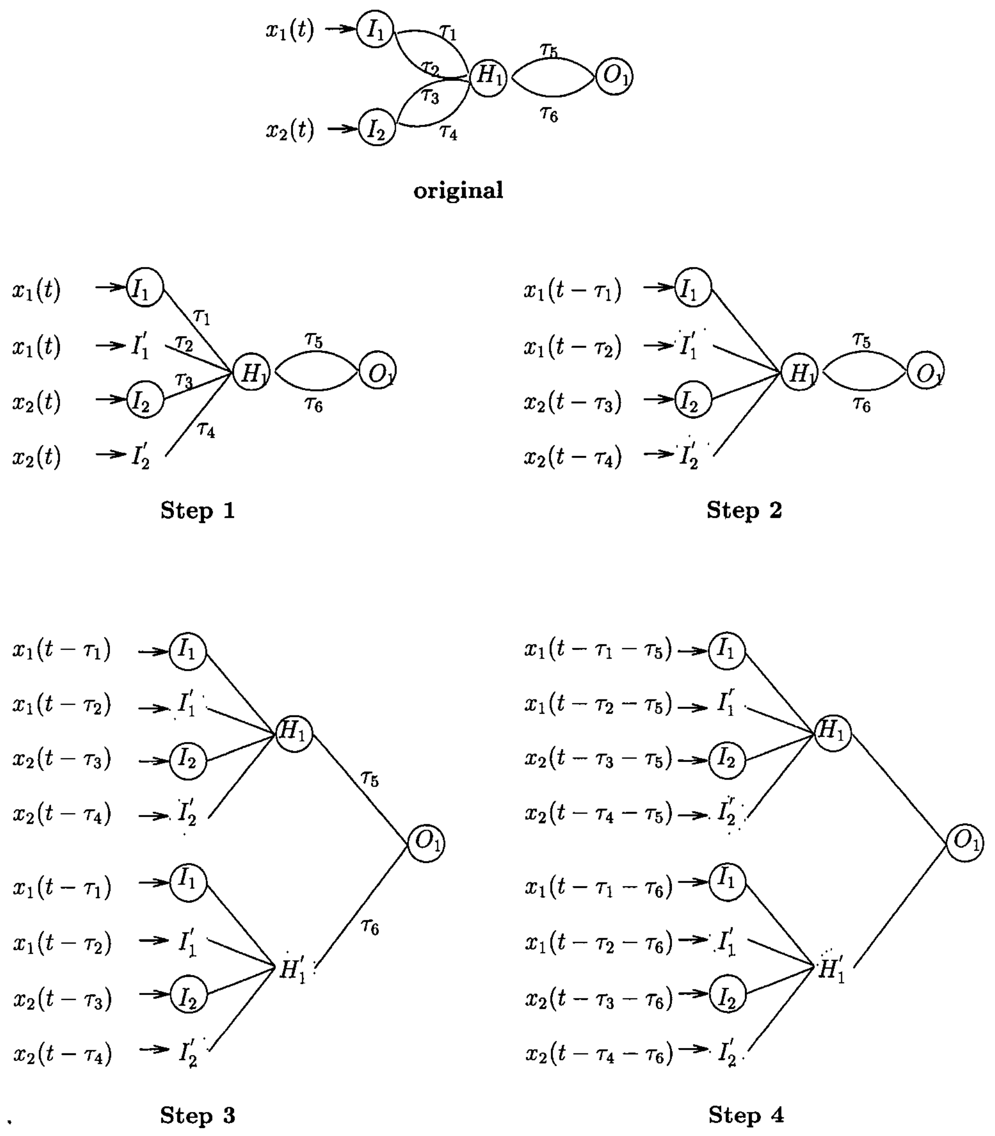

Figure 2: Explicit demonstration of unfold operation.

	original net		unfolded net	
# of output nodes	n_O	$O(n)$	n_O	$O(n)$
# of connections b/t each H/O node pair	d_2	$O(1)$	1	$O(1)$
# of hidden nodes	n_H	$O(n)$	$n_H d_2 n_O$	$O(n^2)$
# of connections b/t each I/H node pair	d_1	$O(1)$	1	$O(1)$
# of input nodes	n_I	$O(n)$	$n_I d_1 n_H d_2 n_O$	$O(n^3)$
# of connections to each output node	$n_H d_2$	$O(n)$	$n_H d_2$	$O(n)$
# of connections to each hidden node	$n_I d_1$	$O(n)$	$n_I d_1$	$O(n)$
total # of connections to output layer	$n_H d_2 n_O$	$O(n^2)$	$n_H d_2 n_O$	$O(n^2)$
total # of connections to hidden layer	$n_I d_1 n_H$	$O(n^2)$	$n_I d_1 n_H d_2 n_O$	$O(n^3)$

Table 1: Comparison of the original and unfolded network configuration.

entire branch of the original node H_1. The input time lags of each branch are adjusted to corresponding time-delay τ_5 and τ_6. In **Step 3**, the new input nodes are generated and the entire input layer is unfolded. Then finally in **Step 4**, each input time lags is readjusted.

3 Hidden Units and Single Layer Time-Delay Adaptation

Property 2 The TDNN is equivalent to its unfolded feedforward network with sufficient number of hidden and input units. However, it is usually impractical to have such large number of hidden nodes and input nodes in an unfolded configuration. An example of a three layered network is shown in Table 1, the total number of hidden units in the unfolded network may grow up to $O(n^2)$, while the total number of input nodes may grow up to $O(n^3)$.

Lemma 1 (single layer time-delay adaptation) *Given a TDNN with ATNN algorithm applied to adapt time-delays on only one layer (layer one or layer two) of connections and adapt weights on both layers. Then, after each iteration of training, the time-delay changes*

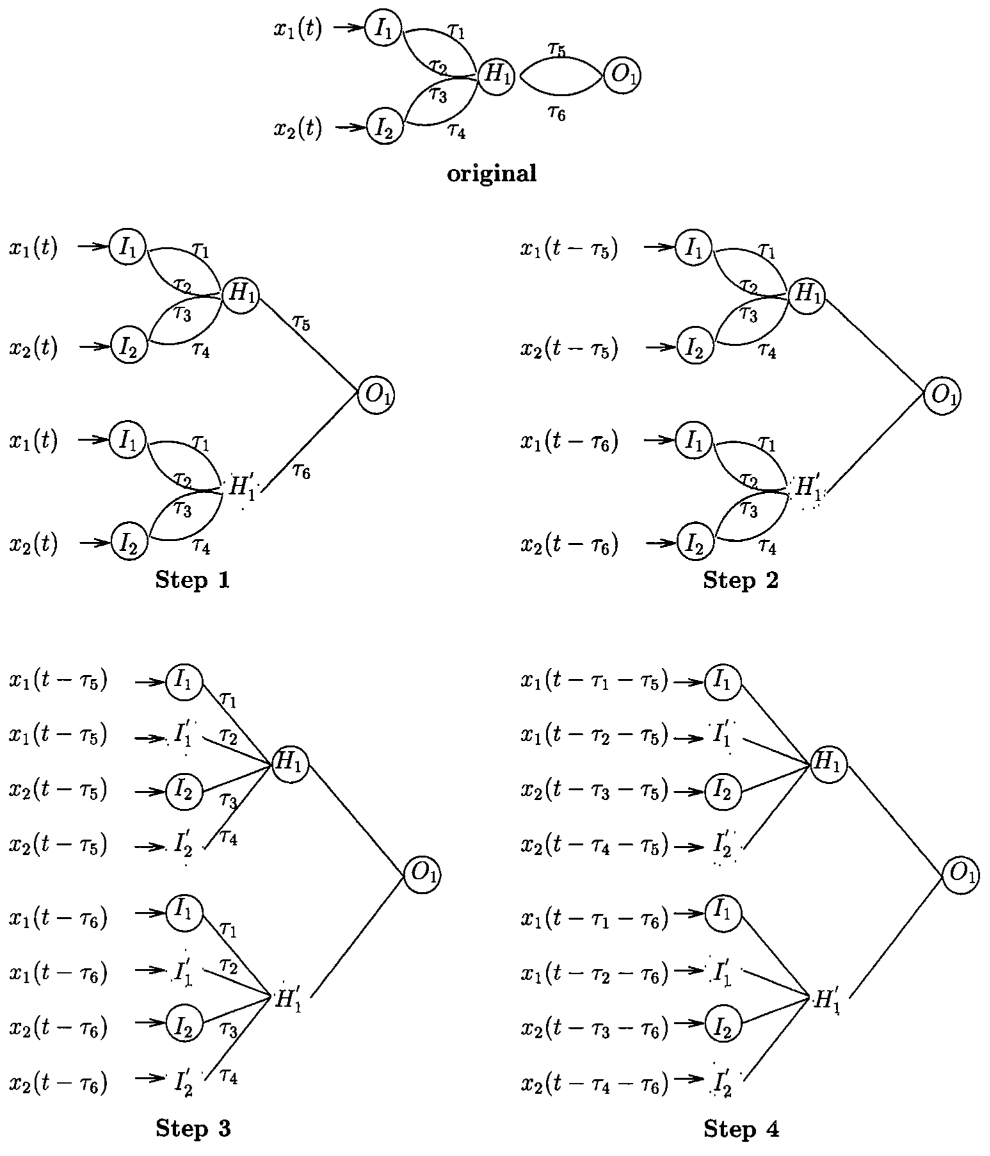

Figure 3: Alternative NUA by unfolding second layer time-delay first.

of the TDNN can be realized in a feedforward network with a sufficiently large number of hidden units $h2^{nd}$, where h and n are the number of hidden and input units of the ATNN respectively, d denotes the sum of the maximum time-delay variables in layer one and layer two of TDNN (i.e., $d = max(T_1) + max(T_2)$, where T_1 and T_2 are time-delay matrices of layer one and two respectively).

Proof: Given n input units and sufficient number of historical time step d, each hidden node is associated with $n \cdot d$ inputs (Property 1). For all $n \cdot d$ elements, there are at most 2^{nd} of combinations (denoted as set $\mathcal{A}$) in terms of whether each element is selected or unselected. Since d is sufficiently large, all the changes in any time-delay elements will fall in the range $[0, d]$.

Given an feedforward net with $h2^{nd}$ hidden units in which each hidden unit of the original ATNN is expanded to 2^{nd} units, such that each of the 2^{nd} units is associated with one of the possible combinations (in set $\mathcal{A}$) of a connection configuration to the historical time steps of all input units.

Applying NUA operation, the ATNN can be unfolded into a feedforward net with the input node taking data $a_i(t - \tau_{layer-one} - \tau_{layer-two})$ out of input node i, where $\tau_{layer-one}$ and $\tau_{layer-two}$ represent the time-delay in layer one and two respectively. For example, $\tau_{layer-one}$ is one of τ_1, τ_2, τ_3, or τ_4 in Figure 2, $\tau_{layer-two}$ is τ_5 or τ_6.

If $\tau_{layer-two}$ is fixed, changing layer one of time-delays is equivalent to making selection of a input set $a_i(t - \tau_{layer-one})$ with a fixed amount of shift $\tau_{layer-two}$. The input data set is a subset of all input data combinations set $\mathcal{A}$.

If $\tau_{layer-one}$ is fixed, changing layer two of time-delays is equivalent to shifting all input data $a_i(t - \tau_{layer-one})$ in the same time step $\tau_{layer-two}$. The shifted data set is also a subset of set $\mathcal{A}$.

Therefore, given $h2^{nd}$ hidden units will be sufficient to represent all different combinations of connection configurations to the historical record of all input nodes. With layer one or layer two of time-delay adapted, the ATNN is equivalent to an unfolded feedforward net with extra and sufficient enough number of hidden units $h2^{nd}$. $\Box$

Given the fact of Theorem 1 and Corollary 1, we need to consider the optimal weights c_i and time-delays $\tau_{i,j,k}$, where the ATNN is applied. Given sufficient number of hidden units and time delays, ATNN with one hidden layer is capable of approximating any spatio-temporal function to any desired degree of accuracy. Practically, for a small number of hidden units, then we need to adapt delays in both layers in order to achieve better mapping or performance.

Remark 1 Given a lower bound number of hidden units, with a proper choice of weights and time-delays, the ATNN can approximate any complex spatiotemporal function to desired accuracy. However, it can get stuck in a local minimum. The convergence is expected to improve if additional hidden units are employed [10, 12].

4 ATNN: Universal Spatiotemporal Function Approximator

The capability of multilayer feedforward networks has been theoretically studied. In previous work, Funahashi, Hornik, Stinchcombe and White have concluded that "standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators" [11]. Similar theorems can also be seen in [8, 16, 4]. The superpositions of a sigmoidal function in achieving universal approximation is also discussed [1].

In this section, we rephrase this mathematical statement and then extend these results to networks with time-delays.

Let $r \in \mathcal{N}$ and A^r be the set of all affine functions from $\mathcal{R}^r$ to $\mathcal{R}$: $A(x) = w \cdot x + b$ where w and x are vectors in $\mathcal{R}^r$, and $b \in \mathcal{R}$ is a scalar. Let $\phi(\cdot)$ be a measurable function from $\mathcal{R}$ to $\mathcal{R}$ and let $\Sigma^r(\phi)$ denote the class of functions $\{f(x) = \Sigma_{i=1}^{q} c_i \phi(A_i(x))\}$ from $\mathcal{R}^r$ to $\mathcal{R}$, where $x \in \mathcal{R}^r, c_i \in \mathcal{R}, A_i \in A^r, q \in \mathcal{N}$. We adopt Funahashi's main results as below:

Theorem 1 (adopted and restated from Funahashi [8]) *Let $\{x_1, ..., x_n\}$ be a set of distinct points of compact set K in Euclidean space $\mathcal{R}^n$ and let $f : \mathcal{R}^n \to \mathcal{R}$ be an arbitrary real valued function on K. If ϕ is a bounded and monotone increasing differentiable function, then for any $\epsilon > 0$, there exists an integer N and real constants c_i, θ_i, and $w_{i,j}$ ($i \in \{1, ..., N\}, j \in \{1, ..., n\}$) such that $\hat{f}(x_1, ...x_n) = \Sigma_{i=1}^{N} c_i \phi(\Sigma_{j=1}^{n} w_{i,j} x_j - \theta_i)$ of $\Sigma^N(\phi)$ satisfies $\forall x \in K, |f(x) - \hat{f}(x)| < \epsilon$. That is, with N hidden units, a three layered network of class $\Sigma(\phi)$ can approximate any function to a desired accuracy.*

Proof of Theorem 1 can be referred to Funahashi (1989) [8] and Hornik, Stinchcombe and White (1989) [11].

Similar theorems can be applied to the TDNN and ATNN to approximate a spatiotemporal function with the extra degrees of freedom in the time-delay domain, considering the network is unfolded. Because the Stone-Weierstrass theorem played a central role in the proof of Theorem 1, we state it here for reference and later use. First we give a few necessary definitions before we introduce the Stone-Weierstrass theorem.

Definition 8 A family $\mathcal{A}$ of real functions defined on a compact set K is an algebra if $\mathcal{A}$ is closed under addition, multiplication and scalar multiplication.

Definition 9 A set $\mathcal{A}$ separates points on compact set K if for every $x, y, x \neq y$ in K, there exists a function f in $\mathcal{A}$ such that $f(x) \neq f(y)$.

Definition 10 A set $\mathcal{A}$ vanishes at no point of K if for each x in K there exists f in $\mathcal{A}$ such that $f(x) \neq 0$.

Theorem 2 (Stone-Weierstrass) *Let $\mathcal{A}$ be an algebra of real continuous functions on a compact set K. If $\mathcal{A}$ separates points on K and if $\mathcal{A}$ vanishes at no point of K, then $\mathcal{A}$ consists of all real continuous functions on K.*

Let $x_1[t], x_2[t], ..., x_n[t]$ be n input signal channels measured/observed in time interval $[t_0, T]$ in a discrete manner. For fixed $\tau_{j,k} \geq 0, j, k \in \mathcal{N}$ and $t - \tau_{j,k}$ in interval $[t_0, T]$, we let $\{< x_1(t-\tau_{1,1}), ..., x_1(t-\tau_{1,d}) >, ..., < x_n(t-\tau_{n,1}), ..., x_n(t-\tau_{n,d}) >\}$ be a set of distinct points of compact set K^d in Euclidean space $\mathcal{R}^{nd}$. We extend Theorem 1 and obtain the following corollary:

Corollary 1 (original contribution) *Let $f : \mathcal{R}^{nd} \to \mathcal{R}$ be an arbitrary real valued function on K^d. If ϕ is a bounded and monotone increasing differentiable function, then for* *any $\epsilon > 0$, there exists integer N, and real constants c_i, θ_i, and $w_{i,j,k}$ ($i \in \{1, ..., N\}, j \in \{1, ..., n\}$), such that*

$$\hat{f}(x) = \Sigma_{i=1}^{N} c_i \phi(\Sigma_{j=1}^{n} \Sigma_{k=1}^{d} w_{i,j,k} x_j(t - \tau_{i,j,k}) - \theta_i)$$

satisfies $\forall x \in K^d, |f(x) - \hat{f}(x)| < \epsilon$. In other words, with at least one hidden layer of N hidden units and d time-delays elements in each input-hidden connections pairs , a three layered TDNN network (with delays employed on the first layer) can approximate any spatiotemporal function to a desired accuracy.

Proof: We apply the Stone-Weierstrass Theorem and use the notations above. Given $\phi(\cdot) : \mathcal{R}^n d \to \mathcal{R}$ and let $\mathcal{A}$ is the set of all affine functions from $\mathcal{R}^{nd}$ to $\mathcal{R}$ such that $\mathcal{A}(x) = w \cdot x + b, x \in K^d$, i.e. $x = \{< x_1(t - \tau_{1,1}), ..., x_1(t - \tau_{1,d}) >, ..., < x_n(t - \tau_{n,1}), ..., x_n(t - \tau_{n,d}) >\}$. We denote $\Sigma^{nd}(\phi)$ as the class of functions $\hat{h}(x) : \{\Sigma c_i \phi(\mathcal{A}(x))\}$. We need to show $\Sigma^{nd}(\phi)$ consists of all real functions on K^d. First we show $\phi(\mathcal{A})$ is separating on K^d that ensures $\Sigma^{nd}(\phi)$ is separating on K^d. Let $K^d \subset \mathcal{R}^{nd}$ be a compact set, so for ϕ from $\mathcal{R}^{nd}$ to $\mathcal{R}$, $\Sigma(\phi)$ is an algebra on K^d. Pick $a, b \in R, a \neq b$ such that $\phi(a) \neq \phi(b)$. Pick $\mathcal{A}(\cdot)$ such that $\mathcal{A}(x) = a$ and $\mathcal{A}(y) = b$, $x, y \in K^d$, then we obtain $\phi(\mathcal{A}(x)) \neq \phi(\mathcal{A}(y))$. Therefore ϕ is separating on K^d.

Secondly, we need to show $\phi(\mathcal{A})$ vanishes at no point on K^d. Pick $b \in \mathcal{R}$ such that $\phi(b) \neq 0$ and set $\mathcal{A}(x) = 0 \cdot x + b$. For all $x \in K^d$, $\phi(\mathcal{A}(X)) = \phi(b)$, the result follows. $\square$

The proof is mathematically equivalent to that of Hornik, Stinchcombe and White's theorem of universal approximation with multilayer feedforward networks. We extend from n dimensional input space to nd dimensional space. In other words, each distinct input point x_i

in n dimensional input space is expanded into a distinct points set $< x_i(t-\tau_1), ..., x_i(t-\tau_d) >$ in nd dimensional input space.

5 Conclusion

The *ATNN* provides more dynamics and flexibility for the network itself to approach an efficient performance level and to optimize its configuration. A network unfolding algorithm (*NUA*) has been explicitly defined to formulate the unfolding operation from *TDNN* or *ATNN* to feedforward network. This procedure provides ways to analyze the complexity of *ATNN*. The *NUA* algorithm also provides a guide line for hardware conversion from conventional feed-forward network without redesign overhead. Extended corollary and lemmas are proposed such that the *ATNN* inherits the properties and capabilities of feedforward network as a universal function approximator.

References

[1] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. on Information Theory*, 39(3):930–945, 1993.

[2] U. Bodenhausen and A. Waibel. The tempo2 algorithm: Adjusting time-delays by supervised learning. In J. E. Moody R. P. Lippmann and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 155–161, Denver 1991, 1991. Morgan Kaufmann, San Mateo.

[3] F. Chapeau-Blondeau and G. Chauvet. Stable, oscillatory, and chaotic regimes in the dynamics of small neural networks with delays. *Neural Networks*, 5:735–743, 1992.

[4] D.S. Chen and R.C. Jain. A robust back propagation learning algorithm for function approximation. *IEEE Trans. on Neural Networks*, 5(3):467–479, May 1994.

[5] S. P. Day and D. S. Camporese. Continuous-time temporal back-propagation. In *International Joint Conference on Neural Networks*, volume 2, pages 95–100, Seattle, 1991. IEEE, New York.

[6] S. P. Day and M. Davenport. Continuous-time temporal back-propagation with adaptive time delays. Neuroprose archive, Ohio State University. Accessible on Internet via anonymous ftp on archive.cis.ohio-state.edu, in pub/neuroprose/day.tempora.ps August, 1991.

[7] S. P. Day and M. R. Davenport. Continuous-time temporal back-propagation with adaptive time delays. *IEEE Trans. on Neural Networks*, 4(2):348–354, March 1993.

[8] K.-I. Funahashi. On the approximate realization of continuous mappings by neural network. *Neural Networks*, 2:183–192, 1989.

[9] P. Haffner and A. Waibel. Multi-state time delay neural networks for continuous speech recognition. In S. J. Hanson J. E. Moody and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 135–142, Denver 1992, 1992. Morgan Kaufmann, San Mateo.

[10] Y. Hirose, K. Yamashita, and S. Hijiya. Back-propagation algorithm which varies the number of hidden units. *Neural Networks*, 4:61–66, 1991.

[11] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[12] D.-T. Lin. *The Adaptive Time-Delay Neural Network: Characterization and Applications to Pattern Recognition, Prediction, and Signal Processing.* PhD thesis, University of Maryland at College Park, 1994.

[13] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Adaptive time-delay neural network for temporal correlation and prediction. In *Intelligent Robots and Computer Vision XI: Biological, Neural Net, and 3-D Methods, Proc. SPIE*, volume 1826, pages 170–181, Boston, November, 1992.

[14] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Trajectory recognition with a time-delay neural network. In *International Joint Conference on Neural Networks*, volume 3, pages 197–202, Baltimore, 1992. IEEE, New York.

[15] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Learning spatiotemporal topology using an adaptive time-delay neural network. In *World Congress on Neural Networks*, volume 1, pages 291–294, Portland, OR, 1993. INNS, New York.

[16] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. IEEE*, 78(9):1481–1497, 1990.

[17] D.W. Tank and J.J. Hopfield. Neural computation by concentrating information in time. *Proceedings of the National Academy of Sciences, USA*, 84:1896–1900, 1987.

[18] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition: Neural networks versus hidden markov models. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages 107–110, April 1988.

[19] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. on Acoust., Speech, Signal Processing*, 37(3):328–339, 1989.

[20] P. J. Werbos. Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78, pages 1550–1560, October 1990.